```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**NTOBLKIO

```
NN      NN  TTTTTTTTTT    000000    BBBBBBBB    LL          KK      KK    IIIIII      000000
NN      NN  TTTTTTTTTT    000000    BBBBBBBB    LL          KK      KK    IIIIII      000000
NN      NN      TT      00      00  BB      BB  LL          KK      KK      II      00      00
NN      NN      TT      00      00  BB      BB  LL          KK      KK      II      00      00
NNNN    NN      TT      00    0000  BB      BB  LL          KK    KK        II      00      00
NNNN    NN      TT      00    0000  BB      BB  LL          KK    KK        II      00      00
NN  NN  NN      TT      00  00  00  BBBBBBBB    LL          KKKKKK          II      00      00
NN  NN  NN      TT      00  00  00  BBBBBBBB    LL          KKKKKK          II      00      00
NN    NNNN      TT      0000    00  BB      BB  LL          KK    KK        II      00      00
NN    NNNN      TT      0000    00  BB      BB  LL          KK    KK        II      00      00
NN      NN      TT      00      00  BB      BB  LL          KK      KK      II      00      00      ....
NN      NN      TT      00      00  BB      BB  LL          KK      KK      II      00      00      ....
NN      NN      TT        000000    BBBBBBBB    LLLLLLLLLL  KK      KK    IIIIII      000000      ....
NN      NN      TT        000000    BBBBBBBB    LLLLLLLLLL  KK      KK    IIIIII      000000      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          $BEGIN  NTOBLKIO,000,NF$NETWORK,<NETWORK BLOCK I/O>
0000     2
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; Facility: RMS
0000    31 ;
0000    32 ; Abstract:
0000    33 ;
0000    34 ;       This module communicates with the File Access Listener (FAL) at the
0000    35 ;       remote node to perform read, write, and space block I/O operations.
0000    36 ;
0000    37 ; Environment: VAX/VMS, executive mode
0000    38 ;
0000    39 ; Author: James A. Krycka,      Creation Date:  18-APR-1978
0000    40 ;
0000    41 ; Modified By:
0000    42 ;
0000    43 ;       V03-004 JAK0145         J A Krycka      12-APR-1984
0000    44 ;               Track changes in DAP message building algorithm.
0000    45 ;
0000    46 ;       V03-003 JAK0122         J A Krycka      22-AUG-1983
0000    47 ;               On $WRITE failure, verify that FAL has sent a Status message
0000    48 ;               before sending a Continue Transfer message to unlock FAL.
0000    49 ;
0000    50 ;       V03-002 JAK0116         J A Krycka      29-JUN-1983
0000    51 ;               Cleanup--remove unused code path.
0000    52 ;
0000    53 ;       V03-001 JAK0104         J A Krycka      22-APR-1983
0000    54 ;               Allow several DATA messages to be blocked in one transmit QIO
0000    55 ;               for $WRITE in file transfer mode.
0000    56 ;
0000    57 ;--
```

```
0000     59          .SBTTL  DECLARATIONS
0000     60
0000     61 ;
0000     62 ; Include Files:
0000     63 ;
0000     64
0000     65          $BDBDEF                         ; Define BDB symbols
0000     66          $DAPPLGDEF                      ; Define DAP prologue symbols
0000     67          $DAPHDRDEF                      ; Define DAP message header
0000     68          $DAPCNFDEF                      ; Define DAP Configuration message
0000     69          $DAPCTLDEF                      ; Define DAP Control message
0000     70          $DAPDATDEF                      ; Define DAP Data message
0000     71          $DAPSTSDEF                      ; Define DAP Status message
0000     72          $IFBDEF                         ; Define IFAB symbols
0000     73          $IRBDEF                         ; Define IRAB symbols
0000     74          $NWADEF                         ; Define Network Work Area symbols
0000     75          $RABDEF                         ; Define Record Access Block symbols
0000     76          $RMSDEF                         ; Define RMS completion codes
0000     77
0000     78 ;
0000     79 ; Macros:
0000     80 ;
0000     81 ;            None
0000     82 ;
0000     83 ; Equated Symbols:
0000     84 ;
0000     85
0000     86          ASSUME  DAP$Q_DCODE_FLG EQ 0
0000     87          ASSUME  NWA$Q_FLG EQ 0
0000     88
0000     89 ;
0000     90 ; Own Storage:
0000     91 ;
0000     92 ;            None
0000     93 ;
```

```
0000     95          .SBTTL  NT$READ - PERFORM NETWORK READ BLOCK FUNCTION
0000     96
0000     97 ;++
0000     98 ; NT$READ - engages in a DAP dialogue with the remote FAL to read the
0000     99 ;       specified blocks.
0000    100 ;
0000    101 ; Calling Sequence:
0000    102 ;
0000    103 ;       BSBW    NT$READ
0000    104 ;
0000    105 ; Input Parameters:
0000    106 ;
0000    107 ;       R4      BDB address
0000    108 ;       R5      VBN of 1st block for transfer
0000    109 ;       R8      RAB address
0000    110 ;       R9      IRAB address
0000    111 ;       R10     IFAB address
0000    112 ;       R11     Impure Area address
0000    113 ;
0000    114 ; Implicit Inputs:
0000    115 ;
0000    116 ;       BDB$L_ADDR
0000    117 ;       BDB$W_NUMB
0000    118 ;       BDB$W_SIZE
0000    119 ;       BDB$L_VBN
0000    120 ;       DAP$L_CRC_RSLT
0000    121 ;       DAP$V_DAPCRC
0000    122 ;       DAP$V_GEQ_V56
0000    123 ;       IFB$V_SQO
0000    124 ;       NWA$V_FTM_EOF
0000    125 ;       NWA$V_FTM_INIT
0000    126 ;       NWA$V_FTM_STORE
0000    127 ;
0000    128 ; Output Parameters:
0000    129 ;
0000    130 ;       R0      Status code (RMS)
0000    131 ;       R1-R3   Destroyed
0000    132 ;       AP      Destroyed
0000    133 ;
0000    134 ; Implicit Outputs:
0000    135 ;
0000    136 ;       BDB buffer contents
0000    137 ;       BDB$W_NUMB
0000    138 ;       BDB$B_REL_VBN destroyed
0000    139 ;       DAP$L_CRC_RSLT
0000    140 ;       NWA$V_FTM_EOF
0000    141 ;       NWA$V_FTM_INIT cleared
0000    142 ;       NWA$V_FTM_RETRV
0000    143 ;       RAB$W_RFA
0000    144 ;
0000    145 ; Completion Codes:
0000    146 ;
0000    147 ;       Standard RMS completion codes
0000    148 ;
0000    149 ; Side Effects:
0000    150 ;
0000    151 ;       None
```

```
                     0000   152 ;
                     0000   153 ;--
                     0000   154
                     0000   155 NT$READ::                                    ; Entry point
                     0000   156                                              ;
   00F0 8F     BB    0006   157          PUSHR   #^M<R4,R5,R6,R7>            ; Save registers
   56   54     D0    000A   158          MOVL    R4,R6                      ; Copy address of BDB
57   3C AA     D0    000D   159          MOVL    IFB$L_NWA_PTR(R10),R7      ; Get address of NWA (and DAP)
     14 A6     B4    0011   160          CLRW    BDB$W_NUMB(R6)             ; Zero # bytes in BDB buffer count
                     0014   161                                             ; Note: BDB$W_NUMB = BDB$W_SIZE on input
     48 A6     94    0014   162          CLRB    BDB$B_REL_VBN(R6)          ; Zero relative VBN to start of buffer
07 67   1B     E0    0017   163          BBS     #NWA$V_FTM_STORE,(R7),10$  ;$READ after $WRITE illegal in FTM
   67   1D     E1    001B   164          BBC     #NWA$V_FTM_EOF,(R7),-      ; Check for EOF received while in FTM
           06       001E   165                  READ_LOOP                  ;  from a previous $READ
        00CB   31    001F   166          BRW     ERREOF                     ; Branch aid
        00C1   31    0022   167 10$:      BRW     ERRFTM                     ; Branch aid
                     0025   168
                     0025   169 ;+
                     0025   170 ; Start of loop to read next block and append it to the user buffer.
                     0025   171 ;
                     0025   172 ; Note: The data access protocol allows only one block to be transferred per
                     0025   173 ;       block I/O request. Therefore, a multi-block user request is performed
                     0025   174 ;       via several one-block DAP requests.
                     0025   175 ;-
                     0025   176
                     0025   177 READ_LOOP:                                   ;
05 6A    2D    E0    0025   178          BBS     #IFB$V_SQO,(R10),10$       ; Branch if sequential-only specified
   51    04    9A    0029   179          MOVZBL  #DAP$K_BLK_VBN,R1          ; Set RAC for DAP message
         0B    11    002C   180          BRB     READ_SEND_CTL              ; Join common code
   67    19    E5    002E   181 10$:      BBCC    #NWA$V_FTM_INIT,(R7),-     ; Branch if no Control message required
         32          0031   182                  READ_BLOCK                 ;  and turn off single-shot flag
                     0032   183          $SETBIT #NWA$V_FTM_RETRV,(R7)      ; Set file transfer mode retrieval flag
   51    05    9A    0036   184          MOVZBL  #DAP$K_BLK_FILE,R1         ; Set RAC for DAP message
                     0039   185
                     0039   186 ;+
                     0039   187 ; Build and send DAP Control message to partner.
                     0039   188 ;-
                     0039   189
                     0039   190 READ_SEND_CTL:                               ;
                     0039   191          $SETBIT #NWA$V_LAST_MSG,(R7)       ; Declare this last message to block
   50    04    D0    003D   192          MOVL    #DAP$K_CTL_MSG,R0          ; Get message type value
      FFBD'    30    0C40   193          BSBW    NT$BUILD_HEAD              ; Construct message header
   85    01    90    0043   194          MOVB    #DAP$K_GET_READ,(R5)+      ; Store CTLFUNC field
   85    03    90    0046   195          MOVB    #<<DAP$M_RAC>!-            ; Store CTLMENU field
                     0049   196                    <DAP$M_KEY>!-            ;
                     0049   197                    0>,(R5)+                 ;
   85    51    90    0049   198          MOVB    R1,(R5)+                   ; Store RAC field
   50    48 A6    9A    004C   199          MOVZBL  BDB$B_REL_VBN(R6),R0       ; Get relative VBN to start of buffer
51   1C A6    50    C1    0050   200          ADDL3   R0,BDB$L_VBN(R6),R1        ; Compute next VBN to request
      FFA8'    30    0055   201          BSBW    NT$CVT_BN4_IMG             ; Store KEY as an image field
      FFA5'    30    0058   202          BSBW    NT$BUILD_TAIL              ; Finish building message
      FFA2'    30    005B   203          BSBW    NT$TRANSMIT                ; Send Control message to FAL
   03 50       E8    005E   204          BLBS    R0,READ_BLOCK              ; Branch on success
      008E     31    0061   205          BRW     EXIT                       ; Branch aid
                     0064   206
                     0064   207 ;+
                     0064   208 ; Receive DAP Data message from partner containing the requested block.
```

```
                     0064   209 ;-
                     0064   210
                     0064   211 READ_BLOCK:                                    ;
                     0064   212         $SETBIT #DAP$K_DAT_MSG,DAP$L_MSG_MASK(R7)
                     0069   213                                                ; Expect response of Data message
          FF94'  30  0069   214         BSBW    NT$RECEIVE                   ; Read block
        5C 50    E9  006C   215         BLBC    R0,CHKEOF                    ; Branch on failure
           15    E1  006F   216         BBC     #DAP$V_DAPCRC,-              ; Branch if partner does not support
     10 28 A7        0071   217                 DAP$Q_SYSCAP(R7),10$         ;  file level CRC checksum
   52   44 A7   7D  0074   218         MOVQ    DAP$Q_FILEDATA(R7),R2        ; Put descriptor of block in <R2,R3>
      0000'CF   0B  0078   219         CRC     W^NT$CRC_TABLE,-             ; Compute CRC (destroying R0-R3)
        20 A7        007C   220                 DAP$L_CRC_RSLT(R7),-         ;  using result of previous CRC
     63    52        007E   221                 R2,(R3)                      ;  calculation as initial CRC value
   20 A7    50   D0  0080   222         MOVL    R0,DAP$L_CRC_RSLT(R7)        ; Store CRC resultant value
   52   44 A7   7D  0084   223 10$:     MOVQ    DAP$Q_FILEDATA(R7),R2        ; Put descriptor of block in <R2,R3>
   50   14 A6   3C  0088   224         MOVZWL  BDB$W_NUMB(R6),R0            ; Get # bytes already in BDB buffer
 51     52   50   A1  008C   225         ADDW3   R0,R2,R1                     ; Compute projected total
   16 A6    51   B1  0090   226         CMPW    R1,BDB$W_SIZE(R6)            ; Will this overflow BDB buffer?
           05   1B  0094   227         BLEQU   20$                          ; Branch if not
52   16 A6    50   A3  0096   228         SUBW3   R0,BDB$W_SIZE(R6),R2         ; Compute # free bytes in BDB buffer
   14 A6    52   A0  009B   229 20$:     ADDW2   R2,BDB$W_NUMB(R6)            ; Update byte count in BDB
     63     52   28  009F   230         MOVC3   R2,(R3),-                    ; Append new block to BDB buffer
        18 B640        00A2   231                 @BDB$L_ADDR(R6)[R0]          ;
                       00A5   232
                       00A5   233 ;+
                       00A5   234 ; Receive DAP Status message from partner if we are not in file transfer mode
                       00A5   235 ; and return record file address of the first block accessed.
                       00A5   236 ;-
                       00A5   237
                       00A5   238 READ_RECV_STS:                               ;
                       00A5   239         RMSSUC                               ; Anticipate success
 12 6A     2D   E0     00A8   240         BBS     #IFB$V_SQO,(R10),CHK1        ; Branch if in file transfer mode
 0E 67     24   E1     00AC   241         BBC     #DAP$V_GEQ_V56,(R7),CHK1; Branch if partner uses DAP before V5.6
                       00B0   242 ; ***** $SETBIT #DAP$K_STS_MSG,DAP$L_MSG_MASK(R7); Implied for receive
          FF4D'  30    00B0   243         BSBW    NT$RECEIVE                   ; Obtain status of read request
        3C 50    E9    00B3   244         BLBC    R0,EXIT                      ; Branch on failure
        48 A6    95    00B6   245         TSTB    BDB$B_REL_VBN(R6)            ; Return RFA value to user RAB on
           03    12    00B9   246         BNEQ    CHK1                         ;  first pass thru loop as RFA refers
           FF42'  30    00BB   247         BSBW    NT$RET_RFA                   ;  to the first block read
                       00BE   248
                       00BE   249 ;
                       00BE   250 ; Determine whether or not user block I/O request has been completed.
                       00BE   251 ;
                       00BE   252
        14 A6    B1    00BE   253 CHK1:    CMPW    BDB$W_NUMB(R6),-             ; Check # bytes received against
        16 A6          00C1   254                 BDB$W_SIZE(R6)               ;  # bytes requested
           2D    1E    00C3   255         BGEQU   EXIT                         ; Branch if user request satisfied
        48 A6    96    00C5   256         INCB    BDB$B_REL_VBN(R6)            ; Update relative VBN for next time thru
           FF5A  31    00C8   257         BRW     READ_LOOP                    ; Branch to read next block
                       00CB   258
                       00CB   259 ;
                       00CB   260 ; Check for end-of-file.
                       00CB   261 ;
                       00CB   262
827A 8F    50   B1     00CB   263 CHKEOF:  CMPW    R0,#<RMS$_EOF&^XFFFF>        ; Is it an end-of-file?
           20   12     00D0   264         BNEQ    EXIT                         ; Branch if not
 06 6A     2D   E1     00D2   265         BBC     #IFB$V_SQO,(R10),10$         ; Branch if not file transfer mode
```

```
                 00D6   266          $SETBIT #NWA$V_FTM_EOF,(R7)     ; Denote that end-of-file has been
      16   11    00DA   267          BRB     EXIT                    ;  reached so that EOF status will be
                 00DC   268                                          ;  returned on next read attempt;
                 00DC   269                                          ;  also it's an input to NT$CLOSE
   14 A6   B5    00DC   270 10$:     TSTW    BDB$W_NUMB(R6)          ; If no data was received from FAL
      11   13    00DF   271          BEQL    EXIT                    ;  then return an EOF condition,
                 00E1   272          RMSSUC                          ;  else return success with the data
      0C   11    00E4   273          BRB     EXIT                    ;  (which will cause BDB$L_VBN to be
                 00E6   274                                          ;  updated on next entry to NT$READ)
                 00E6   275
                 00E6   276 ;+
                 00E6   277 ; Error processing and exit paths for read operation.
                 00E6   278 ;-
                 00E6   279
                 00E6   280 ERRFTM: RMSERR  FTM                     ; Declare file transfer mode error
      05   11    00EB   281          BRB     EXIT                    ;
                 00ED   282 ERREOF: RMSERR  EOF                     ; Declare end-of-file
00F0 8F    BA    00F2   283 EXIT:    POPR    #^M<R4,R5,R6,R7>        ; Restore registers
      05         00F6   284          RSB                             ; Exit with RMS code in R0
```

```
                        .SBTTL  NT$WRITE - PERFORM NETWORK WRITE BLOCK FUNCTION

                ;++
                ; NT$WRITE - engages in a DAP dialogue with the remote FAL to write the
                ;       specified blocks.
                ;
                ; Calling Sequence:
                ;
                ;       BSBW    NT$WRITE
                ;
                ; Input Parameters:
                ;
                ;       R4      BDB address
                ;       R5      VBN of 1st block for transfer
                ;       R8      RAB address
                ;       R9      IRAB address
                ;       R10     IFAB address
                ;       R11     Impure Area address
                ;
                ; Implicit Inputs:
                ;
                ;       BDB buffer contents
                ;       BDB$L_ADDR
                ;       BDB$W_NUMB
                ;       BDB$W_SIZE
                ;       BDB$L_VBN
                ;       DAP$L_CRC_RSLT
                ;       DAP$V_DAPCRC
                ;       DAP$V_GEQ_V56
                ;       IFB$V_SQO
                ;       NWA$V_FTM_INIT
                ;       NWA$V_FTM_RETRV
                ;       NWA$Q_BLD
                ;
                ; Output Parameters:
                ;
                ;       R0      Status code (RMS)
                ;       R1-R3   Destroyed
                ;       AP      Destroyed
                ;
                ; Implicit Outputs:
                ;
                ;       BDB$W_NUMB
                ;       BDB$B_REL_VBN destroyed
                ;       DAP$L_CRC_RSLT
                ;       NWA$V_FTM_INTI cleared
                ;       NWA$V_FTM_STORE
                ;       RAB$W_RFA
                ;
                ; Completion Codes:
                ;
                ;       Standard RMS completion codes
                ;
                ; Side Effects:
                ;
                ;       None
                ;
```

```
                     00F7   343 ;--
                     00F7   344
                     00F7   345 NT$WRITE::                              ; Entry point
                     00F7   346         $TSTPT  NTWRITE                 ;
   00F0 8F    BB     00FD   347         PUSHR   #^M<R4,R5,R6,R7>        ; Save registers
   56   54    D0     0101   348         MOVL    R4,R6                   ; Copy address of BDB
57  3C AA    D0     0104   349         MOVL    IFB$L_NWA_PTR(R10),R7   ; Get address of NWA (and DAP)
   67   1A    E0     0108   350         BBS     #NWA$V_FTM_RETRV,(R7),- ; $WRITE after $READ illegal in FTM
           DA     010B   351                 ERRFTM                  ;
     14 A6    B4     010C   352         CLRW    BDB$W_NUMB(R6)          ; Zero # bytes in BDB buffer count
                     010F   353                                         ; Note: BDB$W_NUMB = BDB$W_SIZE on input
     48 A6    94     010F   354         CLRB    BDB$B_REL_VBN(R6)       ; Zero relative VBN to start of buffer
                     0112   355
                     0112   356 ;+
                     0112   357 ; Start of loop to write next block and append it to the user buffer.
                     0112   358 ;
                     0112   359 ; Note: The data access protocol allows only one block to be transferred per
                     0112   360 ;       block I/O request. Therefore, a multi-block user request is performed
                     0112   361 ;       via several one-block DAP requests.
                     0112   362 ;-
                     0112   363
                     0112   364 WRITE_LOOP:                             ;
05 6A    2D     E0   0112   365         BBS     #IFB$V_SQO,(R10),10$    ; Branch if sequential-only specified
   51    04    9A     0116   366         MOVZBL  #DAP$K_BLK_VBN,R1       ; Set RAC for DAP message
         0B    11     0119   367         BRB     WRITE_SEND_CTL          ; Join common code
   67    19    E5     011B   368 10$:     BBCC    #NWA$V_FTM_INIT,(R7),-  ; Branch if no Control message required
           2E             011E   369                 WRITE_BLOCK             ;  and turn off single-shot flag
                     011F   370         $SETBIT #NWA$V_FTM_STORE,(R7)   ; Set file transfer mode storage flag
   51    05    9A     0123   371         MOVZBL  #DAP$K_BLK_FILE,R1      ; Set RAC for DAP message
                     0126   372
                     0126   373 ;+
                     0126   374 ; Build and send DAP Control message to partner.
                     0126   375 ;-
                     0126   376
                     0126   377 WRITE_SEND_CTL:                          ;
   50    04    D0     0126   378         MOVL    #DAP$K_CTL_MSG,R0       ; Get message type value
      FED4'    30     0129   379         BSBW    NT$BUILD_HEAD           ; Construct message header
   85    04    90     012C   380         MOVB    #DAP$K_PUT_WRITE,(R5)+  ; Store CTLFUNC field
   85    03    90     012F   381         MOVB    #<<DAP$M_RAC>!-         ; Store CTLMENU field
                     0132   382                   <DAP$M_KEY>!-         ;
                     0132   383                   0>,(R5)+              ;
   85    51    90     0132   384         MOVB    R1,(R5)+                ; Store RAC field
   50    48 A6    9A     0135   385         MOVZBL  BDB$B_REL_VBN(R6),R0    ; Get relative VBN to start of buffer
51  1C A6    50    C1     0139   386         ADDL3   R0,BDB$L_VBN(R6),R1     ; Compute next VBN to request
      FEBF'    30     013E   387         BSBW    NT$CVT_BR4_IMG          ; Store KEY as an image field
      FEBC'    30     0141   388         BSBW    NT$BUILD_TAIL           ; Finish building message
      FEB9'    30     0144   389         BSBW    NT$TRANSMIT             ; Send Control message to FAL
   03  50    E8     0147   390         BLBS    R0,WRITE_BLOCK          ; Branch on success
     00BB    31     014A   391         BRW     EXIT1                   ; Branch on failure
                     014D   392
                     014D   393 ;+
                     014D   394 ; Build and send DAP Data message to partner containing the next block.
                     014D   395 ;-
                     014D   396
                     014D   397 WRITE_BLOCK:                            ;
09 6A    2D     E1   014D   398         BBC     #IFB$V_SQO,(R10),5$     ; Branch if not in file transfer mode
   00CA C7    B1     0151   399         CMPW    NWA$W_DAPBUFSIZ(R7),-   ; Allow blocking of DATA messages in
```

```
                0410 8F         0155   400                      #<1024+16>                  ;  transmit QIO if at least two will
                     04   1E    0158   401              BGEQU   10$                         ;  fit in the DAP buffer
                          15    015A   402  5$:         $SETBIT #NWA$V_LAST_MSG,(R7)        ; Declare this last message to block
                50   08   D0    015E   403  10$:        MOVL    #DAP$K_DAT_MSG,R0           ; Get message type value
                   FE9C'  30     0161   404              BSBW    NT$BUILD_HEAD               ; Construct message header
           54   00F4 C7   D0     0164   405              MOVL    NWA$Q_BLD+4(R7),R4          ; Get address of build message buffer
      50   48 A6   9A     0169   406              MOVZBL  BDB$B_REL_VBN(R6),R0        ; Get relative VBN to start of buffer
51    1C A6   50   C1     016D   407              ADDL3   R0,BDB$L_VBN(R6),R1         ; Compute next VBN to request
                   FE8B'  30     0172   408              BSBW    NT$CVT_BN4_IMG              ; Store RECNUM as an image field
           53   55   D0     0175   409              MOVL    R5,R3                       ; Save next byte pointer
      50   14 A6   3C     0178   410              MOVZWL  BDB$W_NUMB(R6),R0           ; Get # bytes already sent from BDB buf
52    16 A6   50   A3     017C   411              SUBW3   R0,BDB$W_SIZE(R6),R2        ; Compute # bytes remaining to send
           0200 8F   52   B1     0181   412              CMPW    R2,#512                     ; Is it more than one block?
                     05   1B    0186   413              BLEQU   20$                         ; Branch if not
           52   0200 8F   B0     0188   414              MOVW    #512,R2                     ; Send exactly one block
              14 A6   52   A0     018D   415  20$:       ADDW2   R2,BDB$W_NUMB(R6)           ; Update byte count in BDB for next time
           51   55   54   C3     0191   416              SUBL3   R4,R5,R1                    ; Compute # DAP overhead bytes in msg
           55   51   52   C1     0195   417              ADDL3   R2,R1,R5                    ; Compute projected size of DAP message
           00CA C7   55   B1     0199   418              CMPW    R5,NWA$W_DAPBUFSIZ(R7)      ; Make sure message will fit in buffer
                     63   1A    019E   419              BGTRU   ERRRSZ                      ; Branch if record is too big
           0120 C7   52   7D     01A0   420              MOVQ    R2,NWA$Q_SAVE_DESC(R7)      ; Save descriptor of user block
                     24   BB    01A5   421              PUSHR   #^M<R2,R5>                  ; Save registers
63         18 B640   52   28     01A7   422              MOVC3   R2,@BDB$L_ADDR(R6)[R0],(R3); Move block into DAP message
                     24   BA    01AD   423              POPR    #^M<R2,R5>                  ; Restore registers
           55   53   D0     01AF   424              MOVL    R3,R5                       ; Save next byte pointer
                   FE4B'  30     01B2   425              BSBW    NT$BUILD_TAIL               ; Finish building message
                     15   E1    01B5   426              BBC     #DAP$V_DAPCRC,-             ; Branch if partner does not support
              11 28 A7          01B7   427                      DAP$Q_SYSCAP(R7),30$        ;  file level CRC checksum
           52   0120 C7   7D     01BA   428              MOVQ    NWA$Q_SAVE_DESC(R7),R2      ; Put descriptor of block in <R2,R3>
                0000'CF   0B     01BF   429              CRC     W^NT$CRC_TABLE,-            ; Compute CRC (destroying R0-R3)
                     20 A7          01C3   430                      DAP$L_CRC_RSLT(R7),-        ;  using result of previous CRC
           63   52                01C5   431                      R2,(R3)                     ;  calculation as initial CRC value
      20 A7   50   D0     01C7   432              MOVL    R0,DAP$L_CRC_RSLT(R7)       ; Store CRC resultant value
           FE32'  30     01CB   433  30$:         BSBW    NT$TRANSMIT                 ; Write block
      23 50   E9          01CE   434              BLBC    R0,CHKSTS                   ; Branch on failure
                          01D1   435
                          01D1   436 ;+
                          01D1   437 ; Receive DAP Status message from partner if we are not in file transfer mode
                          01D1   438 ; and return record file address of the first block accessed.
                          01D1   439 ;-
                          01D1   440
                          01D1   441 WRITE_RECV_STS:                                      ;
12 6A   2D   E0           01D1   442              BBS     #IFB$V_SQO,(R10),CHK2       ; Branch if in file transfer mode
0E 67   24   E1           01D5   443              BBC     #DAP$V_GEQ_V56,(R7),CHK2; Branch if partner uses DAP before V5.6
                          01D9   444 ; ***** $SETBIT #DAP$K_STS_MSG,DAP$L_MSG_MASK(R7); Implied for receive
           FE24'  30      01D9   445              BSBW    NT$RECEIVE                  ; Obtain status of write request
      15 50   E9          01DC   446              BLBC    R0,CHKSTS                   ; Branch on failure
      48 A6   95          01DF   447              TSTB    BDB$B_REL_VBN(R6)           ; Return RFA value to user RAB on
                03   12    01E2   448              BNEQ    CHK2                        ;  first pass thru loop as RFA refers
           FE19'  30      01E4   449              BSBW    NT$RET_RFA                  ;  to the first block written
                          01E7   450
                          01E7   451 ;
                          01E7   452 ; Determine whether or not user block I/O request has been completed.
                          01E7   453 ;
                          01E7   454
           14 A6   B1     01E7   455 CHK2:    CMPW    BDB$W_NUMB(R6),-            ; Check # bytes transmitted against
           16 A6          01EA   456                     BDB$W_SIZE(R6)              ;  # bytes requested
```

```
                1A   1E  01EC    457                 BGEQU   EXIT1                   ; Branch if user request satisfied
          48 A6   96  01EE    458                 INCB    BDB$B_REL_VBN(R6)       ; Update relative VBN for next time thru
           FF:E   31  01F1    459                 BRW     WRITE_LOOP              ; Branch to write next block
                      01F4    460
                      01F4    461 ;+
                      01F4    462 ; Error processing and exit paths for write operation.
                      01F4    463 ;-
                      01F4    464
          30 A7   91  01F4    465 CHKSTS:         CMPB    DAP$B_TYPE(R7),-        ; Branch if failure was not the result
                09            01F7    466                 #DAP$K_STS_MSG          ;  of Status message returned by FAL
                0E  12  01F8    467                 BNEQ    EXIT1                   ;
                01  BB  01FA    468                 PUSHR   #^M<R0>                 ; Save primary error code
           FE01'    30  01FC    469                 BSBW    NT$RESUME_FAL           ; Tell FAL what to do on write error via
                      01FF    470                                                 ;  interrupt Continue Transfer message
                01  BA  01FF    471                 POPR    #^M<R0>                 ; Restore primary error code
                05  11  0201    472                 BRB     EXIT1                   ;
                      0203    473 ERRRSZ:         RMSERR  RSZ                     ; Invalid record size
        00F0 8F   BA  0208    474 EXIT1:          POPR    #^M<R4,R5,R6,R7>        ; Restore registers
                05  020C    475                 RSB                             ; Exit with RMS code in R0
```

```
                     020D   477          .SBTTL  NT$SPACE - PERFORM NETWORK SPACE BLOCK FUNCTION
                     020D   478
                     020D   479 ;++
                     020D   480 ; NT$SPACE - engages in a DAP dialogue with the remote FAL to space the
                     020D   481 ;       file forward or backward the specified number of blocks.
                     020D   482 ;
                     020D   483 ; Calling Sequence:
                     020D   484 ;
                     020D   485 ;       BSBW    NT$SPACE
                     020D   486 ;
                     020D   487 ; Input Parameters:
                     020D   488 ;
                     020D   489 ;       R1      # blocks to space as a signed number
                     020D   490 ;       R8      RAB address
                     020D   491 ;       R9      IRAB address
                     020D   492 ;       R10     IFAB address
                     020D   493 ;       R11     Impure Area address
                     020D   494 ;
                     020D   495 ; Implicit Inputs:
                     020D   496 ;
                     020D   497 ;       None
                     020D   498 ;
                     020D   499 ; Output Parameters:
                     020D   500 ;
                     020D   501 ;       R0      Status code (RMS)
                     020D   502 ;       R1-R5   Destroyed
                     020D   503 ;       R6      Actual # blocks spaced as an unsigned number
                     020D   504 ;       R7      Destroyed
                     020D   505 ;       AP      Destroyed
                     020D   506 ;
                     020D   507 ; Implicit Outputs:
                     020D   508 ;
                     020D   509 ;       None
                     020D   510 ;
                     020D   511 ; Completion Codes:
                     020D   512 ;
                     020D   513 ;       Standard RMS completion codes
                     020D   514 ;
                     020D   515 ; Side Effects:
                     020D   516 ;
                     020D   517 ;       None
                     020D   518 ;
                     020D   519 ;--
                     020D   520
                     020D   521 NT$SPACE::                              ; Entry point
                     020D   522          $TSTPT  NTSPACE                ;
          56   D4    0213   523          CLRL    R6                     ; Zero # blocks spaced
    34 6A   2D   E0  0215   524          BBS     #IFB$V_SQO,(R10),ERRFTM2; Network space function not allowed
                     0219   525                                         ;  if file transfer mode selected
    57   3C AA   D0  0219   526          MOVL    IFB$L_NWA_PTR(R10),R7  ; Get address of NWA (and DAP)
                     021D   527
                     021D   528 ;+
                     021D   529 ; Build and send DAP Control message to partner.
                     021D   530 ;-
                     021D   531
                     021D   532 SPACE_SEND_CTL:                         ;
                     021D   533          $SETBIT #NWA$V_LAST_MSG,(R7)   ; Declare this last message to block
```

```
                50    04   D0  0221   534          MOVL    #DAP$K_CTL_MSG,R0          ; Get message type value
                   FDD9'   30  0224   535          BSBW    NT$BUILD_HEAD              ; Construct message header
                     51    D5  0227   536          TSTL    R1                         ; Space foreward request?
                     05    19  0229   537          BLSS    10$                        ; Branch if not
                85   11    90  022B   538          MOVB    #DAP$K_SPACE_FW,(R5)+      ; Set CTLFUNC field for forward space
                     06    11  022E   539          BRB     20$                        ;
                85   12    90  0230   540 10$:     MOVB    #DAP$K_SPACE_BW,(R5)+      ; Set CTLFUNC field for backward space
                51   51    CE  0233   541          MNEGL   R1,R1                      ; Make value positive
                85   02    90  0236   542 20$:     MOVB    #DAP$M_KEY,(R5)+           ; Store CTLMENU field
                   FDC4'   30  0239   543          BSBW    NT$CVT_BN4_IMG             ; Store KEY as an image field
                   FDC1'   30  023C   544          BSBW    NT$BUILD_TAIL              ; Finish building message
                   FDBE'   30  023F   545          BSBW    NT$TRANSMIT                ; Send Control message to FAL
                 0D 50     E9  0242   546          BLBC    R0,EXIT2                   ; Branch on failure
                           0245   547
                           0245   548 ;+
                           0245   549 ; Receive DAP Status message from partner to obtain actual number of blocks
                           0245   550 ; spaced.
                           0245   551 ;-
                           0245   552
                           0245   553 SPACE_RECV_STS:                               ;
                           0245   554 ; ***** $SETBIT #DAP$K_STS_MSG,DAP$L_MSG_MASK(R7); Implied for receive
                           0245   555                                               ; Expect response of Status message
                   FDB8'   30  0245   556          BSBW    NT$RECEIVE                 ; Receive status of space request
             56    48 A7   D0  0248   557          MOVL    DAP$L_RECNUM2(R7),R6       ; Get # blocks actually spaced
                           024C   558                                               ;  as an unsigned number
                     05    024C   559          RSB                                ; Exit with RMS code in R0
                           024D   560
                           024D   561 ;+
                           024D   562 ; Error processing and exit paths for space operation.
                           024D   563 ;-
                           024D   564
                           024D   565 ERRFTM2:RMSERR  FTM                           ; Declare file transfer mode error
                        05 0252   566 EXIT2:  RSB                                 ; Exit with RMS code in R0
                           0253   567
                           0253   568          .END                               ; End of module
```

```
$$.PSECT_EP                     =  00000000
$$RMSTEST                       =  0000001A
$$RMS_PBUGCHK                   =  00000010
$$RMS_TBUGCHK                   =  00000008
$$RMS_UMODE                     =  00000004
BDB$B_REL_VBN                   =  00000048
BDB$L_ADDR                      =  00000018
BDB$L_VBN                       =  0000001C
BDB$W_NUMB                      =  00000014
BDB$W_SIZE                      =  00000016
CHK1                               000000BE  R       01
CHK2                               000001E7  R       01
CHKEOF                             000000CB  R       01
CHKSTS                             000001F4  R       01
DAP$B_BITCNT                       00000035
DAP$B_BLKCNT                       00000056
DAP$B_CTLFUNC                      00000040
DAP$B_DCODE_FID                    00000019
DAP$B_DCODE_MAC                    0000001B
DAP$B_DCODE_MSG                    0000001A
DAP$B_DECVER                       00000047
DAP$B_ECONUM                       00000045
DAP$B_FILESYS                      00000043
DAP$B_FLAGS                        00000031
DAP$B_KRF                          00000047
DAP$B_LEN256                       00000034
DAP$B_LENGTH                       00000033
DAP$B_OSTYPE                       00000042
DAP$B_RAC                          00000046
DAP$B_STREAMID                     00000032
DAP$B_TYPE                         00000030
DAP$B_USRNUM                       00000046
DAP$B_USRVER                       00000048
DAP$B_VERNUM                       00000044
DAP$B_X_FIELD                      00000024
DAP$C_BLN                          000000C0
DAP$K_BLK_FILE                  =  00000005
DAP$K_BLK_VBN                   =  00000004
DAP$K_BLN                          000000C0
DAP$K_CTL_MSG                   =  00000004
DAP$K_DAT_MSG                   =  00000008
DAP$K_GET_READ                  =  00000001
DAP$K_PUT_WRITE                 =  00000004
DAP$K_SEQ_ACC                   =  00000000
DAP$K_SPACE_BW                  =  00000012
DAP$K_SPACE_FW                  =  00000011
DAP$K_STS_MSG                   =  00000009
DAP$L_CMWA                         00000030
DAP$L_CRC_RSLT                     00000020
DAP$L_DCODE_STS                    00000018
DAP$L_MSG_MASK                     0000001C
DAP$L_RECNUM1                      00000040
DAP$L_RECNUM2                      00000048
DAP$L_ROP                          00000050
DAP$L_SSPWA                        00000080
DAP$L_STV                          0000004C
DAP$L_TEMP                         00000090
DAP$M_BITCNT                    =  00000008
DAP$M_BLKCNT                    =  00000040
DAP$M_KEY                       =  00000002
DAP$M_RAC                       =  00000001
DAP$M_SEGMENT                   =  00000040
DAP$M_TMP1$                     =  00000008
DAP$M_TMP2$                     =  FFF80000
DAP$Q_DCODE_FLG                    00000000
DAP$Q_FILEDATA                     00000044
DAP$Q_KEY                          00000048
DAP$Q_MSG_BUF1                     00000008
DAP$Q_MSG_BUF2                     00000010
DAP$Q_STX                          00000050
DAP$Q_SYSCAP                       00000028
DAP$Q_SYSPEC                       00000038
DAP$V_DAPCRC                    =  00000015
DAP$V_GEQ_V56                   =  00000024
DAP$W_BUFSIZ                       00000040
DAP$W_CTLMENU                      00000044
DAP$W_DISPLAY2                     00000054
DAP$W_PARTNER                      00000006
DAP$W_RFA                          00000042
DAP$W_STSCODE                      00000040
DAP$W_VERSION                      00000004
ERREOF                             000000ED  R       01
ERRFTM                             000000E6  R       01
ERRFTM2                            0000024D  R       01
ERRRSZ                             00000203  R       01
EXIT                               000000F2  R       01
EXIT1                              00000208  R       01
EXIT2                              00000252  R       01
IFB$L_NWA_PTR                   =  0000003C
IFB$V_SQO                       =  0000002D
NT$BUILD_HEAD                      ********    X     01
NT$BUILD_TAIL                      ********    X     01
NT$CRC_TABLE                       ********    X     01
NT$CVT_BN4_IMG                     ********    X     01
NT$READ                            00000000  RG      01
NT$RECEIVE                         ********    X     01
NT$RESUME_FAL                      ********    X     01
NT$RET_RFA                         ********    X     01
NT$SPACE                           0000020D  RG      01
NT$TRANSMIT                        ********    X     01
NT$WRITE                           000000F7  RG      01
NWA$B_ALLXABCNT                    0000011C
NWA$B_DAP_RAC                      000000C9
NWA$B_FILESYS                      000000C5
NWA$B_KEYXABCNT                    0000011D
NWA$B_NETSTRSIZ                    0000016F
NWA$B_NODBUFSIZ                    00000168
NWA$B_ORG                          000000C6
NWA$B_OSTYPE                       000000C4
NWA$B_RFM                          000000C7
NWA$B_RMS_RAC                      000000C8
NWA$C_BLN                          00000800
NWA$K_BLN                          00000800
NWA$L_ALLXABADR                    00000100
```

```
NWA$L_DATXABADR          00000104
NWA$L_DEV                000000C0
NWA$L_FHCXABADR          00000108
NWA$L_KEYXABADR          0000010C
NWA$L_MSG_MASK           000000D4
NWA$L_PROXABADR          00000110
NWA$L_RDTXABADR          00000114
NWA$L_SAVE_FLGS          00000128
NWA$L_SUMXABADR          00000118
NWA$L_THREAD             000000FC
NWA$L_XLTATTR            00000238
NWA$L_XLTBUFFLG          0000022C
NWA$L_XLTCNT             00000228
NWA$L_XLTMAXINDX         00000234
NWA$L_XLTSIZ             00000230
NWA$Q_ACS                00000244
NWA$Q_BIGBUF             00000170
NWA$Q_BLD                000000F0
NWA$Q_FLG                00000000
NWA$Q_INODE              0000025C
NWA$Q_IOSB               000000D8
NWA$Q_LNODE              00000160
NWA$Q_LOGNAME            0000023C
NWA$Q_NCB                00000264
NWA$Q_RCV                000000E0
NWA$Q_SAVE_DESC          00000120
NWA$Q_XLTBUF1            0000024C
NWA$Q_XLTBUF2            00000254
NWA$Q_XMT                000000E8
NWA$T_ACSBUF             0000026C
NWA$T_AUXBUF             000005E0
NWA$T_DAP                00000000
NWA$T_INODEBUF           000004AC
NWA$T_ITM_ATTR           00000200
NWA$T_ITM_END            00000224
NWA$T_ITM_LST            00000200
NWA$T_ITM_MAXINDX        00000218
NWA$T_ITM_STRING         0000020C
NWA$T_NCBBUF             0000052C
NWA$T_NODEBUF            00000169
NWA$T_RCVBUF             000001A0
NWA$T_SCAN               00000100
NWA$T_TEMP               00000120
NWA$T_XLTBUF1            000002AC
NWA$T_XLTBUF2            000003AC
NWA$T_XMTBUF             000003C0
NWA$V_FTM_EOF          = 0000001D
NWA$V_FTM_INIT         = 00000019
NWA$V_FTM_RETRV        = 0000001A
NWA$V_FTM_STORE        = 0000001B
NWA$V_LAST_MSG         = 00000000
NWA$W_BUILD              000000D2
NWA$W_DAPBUFSIZ          000000CA
NWA$W_DIR_OFF            000000CC
NWA$W_DISPLAY            000000D0
NWA$W_FIL_OFF            000000CE
NWA$W_JNLXABJOP          0000011E
PIO$A_TRACE                ********    X   01
READ_BLOCK                 00000064 R      01
READ_LOOP                  00000025 R      01
READ_RECV_STS              000000A5 R      01
READ_SEND_CTL              00000039 R      01
RMS$_EOF               = 0001827A
RMS$_FTM               = 000187C4
RMS$_RSZ               = 000186A4
SPACE_RECV_STS             00000245 R      01
SPACE_SEND_CTL             0000021D R      01
TPT$L_NTREAD               ********      X 01
TPT$L_NTSPACE              ********      X 01
TPT$L_NTWRITE              ********      X 01
WRITE_BLOCK                0000014D R      01
WRITE_LOOP                 00000112 R      01
WRITE_RECV_STS             000001D1 R      01
WRITE_SEND_CTL             00000126 R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
NF$NETWORK	00000253 (595.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000800 (2048.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.09	00:00:00.75
Command processing	114	00:00:00.67	00:00:03.64
Pass 1	342	00:00:12.96	00:00:29.71
Symbol table sort	0	00:00:01.69	00:00:02.93
Pass 2	111	00:00:02.51	00:00:06.19
Symbol table output	23	00:00:00.17	00:00:00.81
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	626	00:00:18.12	00:00:44.07

The working set limit was 1350 pages.
68178 bytes (134 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1202 non-local and 19 local symbols.
568 source lines were read in Pass 1, producing 15 object records in Pass 2.
27 pages of virtual memory were used to define 26 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	18
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	22

1418 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:NTOBLKIO/OBJ=OBJ$:NTOBLKIO MSRC$:NTOBLKIO/UPDATE=(ENH$:NTOBLKIO)+LIB$:RMS/LIB
```